Terbit online pada laman web jurnal: **http://journal.isas.or.id/index.php/JASENS**

JOURNAL OF APPLIED SMART ELECTRICAL NETWORK AND SYSTEMS (JASENS)

Vol. 1 No. 2 (2020) 70 - 75 ISSN Media Elektronik: 2723-5467

Antarmuka Mikrokontroller IoT (ESP32) Dengan USB Host max3421e

Rieke Adriati Wijayanti[1], Ahmad Wilda Yulianto[2], Dianthy Marya[3], M. Syirajuddin S.[4], Nurul Hidayati[5]
[1,2,3,4,5]Jurusan Teknik Elektro, Politeknik Negeri Malang
[1]riekeaw@polinema.ac.id*, [2]ahmadwildan@polinema.ac.id, [3]dianthy@polinema.ac.id, [4]syirajuddin@polinema.ac.id, [5]nurulhid8@polinema.ac.id,

Abstract

Electronic equipment made using old technology or electronic equipment in the entry level category has not been supported by networking equipment, so for the data communication process, the microcontroller requires interfacing facilities that are in accordance with the electronic equipment used, such as a USB port. With the microcontroller that supports IoT, it allows electronic equipment to communicate over the network. An IoT microcontroller such as the ESP32 is equipped with a WiFi feature but is not equipped with a USB controller feature, while the USB Host max3421e supports the communication process using SPI, so that those two microcontrollers can be used to form an interface using the SPI bus. This interface can be applied to electronic equipment with old technology and entry level electronic equipment for wireless communication. For the needs of making an interface between the ESP32 and max3421e, a software was developed by analyzing the SPI features of the ESP32 and the USB protocol according to the USB device state diagram. The results obtained are the handshake process between systems developed with USB devices in the Low-Speed and Full-Speed categories such as printers, flashdisk, bluetooth mouse and external hard disk, and the device descriptor data of each device tested can be read properly.

Keywords: interface, microcontroller ESP32, USB Host max3421e, SPI, handshake

Abstrak

Peralatan elektronik yang dibuat menggunakan teknologi lama maupun peralatan elektronik dengan kategori *entry level* belum didukung oleh peralatan *networking*, sehingga untuk proses komunikasi data, mikrokontroller memerlukan fasilitas *interfacing* yang sesuai dengan peralatan elektronik yang digunakan, seperti USB *port*. Dengan adanya mikrokontroller yang mendukung IoT, memungkinkan peralatan elektronik tersebut untuk melakukan komunikasi melalui *network*. Mikrokontroller IoT seperti ESP32 telah dilengkapi dengan fitur WiFi namun tidak dilengkapi dengan fitur USB *controller*, sedangkan USB *Host* max3421e mendukung proses komunikasi menggunakan SPI, sehingga kedua mikrokontroller tersebut dapat dimanfaatkan untuk membentuk antarmuka menggunakan SPI *bus*. Antarmuka tersebut dapat diaplikasikan pada peralatan elektronik dengan teknologi lama dan peralatan elektronik *entry level* untuk melakukan komunikasi secara *wireless*. Untuk kebutuhan pembuatan antarmuka antara ESP32 dan max3421e, dikembangkan *software* dengan cara menganalisis fitur SPI dari ESP32 dan protokol USB sesuai dengan *device state diagram* dari USB. Hasil yang didapatkan adalah proses *handshake* antara sistem yang dikembangkan dengan device USB dengan kategori Low-Speed dan Full-Speed seperti printer, flashdisk, *bluetooth mouse* dan *external hard disk* dapat berjalan dengan baik, serta data *device descriptor* dari setiap *device* yang diuji dapat dibaca dengan benar.

Kata kunci: antarmuka, mikrokontroller ESP32, USB *Host* max3421e, SPI, *handshake*

Diterima Redaksi : 01-12-2020 | Selesai Revisi : 18-12-2020 | Diterbitkan Online : 31-12-2020

1. Pendahuluan

Peralatan elektronik terkini sangat lekat dengan fasilitas *networking* seperti WiFi sehingga proses komunikasinya dapat dilakukan melalui WiFi. Sedangkan peralatan elektronik yang dibuat menggunakan teknologi lama maupun peralatan elektronik dengan kategori *entry level* belum didukung oleh peralatan *networking*, sehingga untuk proses komunikasi data, mikrokontroller memerlukan fasilitas *interfacing* yang sesuai dengan peralatan elektronik yang digunakan, seperti USB *port*. Namun dengan adanya mikrokontroller yang mendukung IoT, memungkinkan peralatan elektronik dengan teknologi lama maupun peralatan elektronik dengan kategori *entry level* untuk melakukan komunikasi melalui *network* [1].

Mikrokontroller IoT seperti ESP32 memiliki fitur WiFi tetapi tidak memiliki fitur USB *controller* yang terintegrasi dalam 1 *chip*. Sedangkan USB Host max3421e mendukung proses komunikasi menggunakan SPI (Serial Peripheral Interface), sehingga kedua mikrokontroller tersebut dapat saling

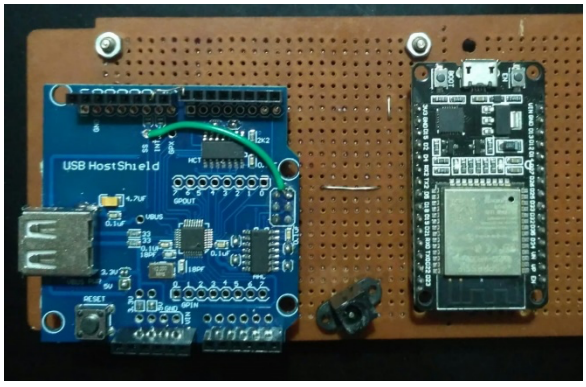
dimanfaatkan untuk proses antarmuka menggunakan SPI *bus* [2].

Pada penelitian terdahulu yang telah dilakukan, USB *Host* max3421e dihubungkan dengan modul mikrokontroller Arduino yang memiliki fasilitas terbatas menggunakan SPI bus. Kemudian komunikasi dengan USB *device* dilakukan secara *wired*. Konfigurasi tersebut digunakan untuk menerima sinyal kontrol dan sensorik melalui USB *port* dari pergerakan *quadcopter* untuk kemudian dijalankan dari *smartphone* [3] dan juga diaplikasikan pada sistem penginderaan optic portable untuk mendeteksi sepktrum fluoresensi secara cepat [4]. Pada penelitian lain mengenai pengambilan gambar pada kamera DSLR, USB Host max3421e juga diaplikasikan sebagai perangkat yang menjembatani komunikasi antara mikrokontroller Arduino dengan kamera DSLR [5]. Selain itu, penelitian lain yang dilakukan oleh Novriadi adalah memanfaatkan mikrokontroller Arduino Mega yang dikombinasikan dengan USB Host max3421e untuk mengatur *overhead crane* baik secara *wired* maupun *wireless* dengan menggunakan Bluetooth [6].

Pada penelitian ini, akan dibangun sebuah sistem dengan menggunakan mikrokontroller ESP32 yang memiliki fasilitas *networking* (WiFi), 32 bit *processor*, RAM yang lebih besar dibandingkan Arduino [7][8] agar dapat memfasilitasi aplikasi yang membutuhkan transaksi data yang lebih besar, seperti transfer data yang dikirimkan ke printer. Sehingga proses komunikasi yang selama ini hanya bisa dilakukan melalui *wired* (kabel USB), akan dapat dilakukan juga melalui *wireless* (WiFi). Namun pada penulisan jurnal ini masih dibatasi pada proses *handshaking* antara USB Host dengan *device*.

2. Metode Penelitian

Penelitian ini diawali dengan merangkai mikrokontroller ESP32 dan USB *Host* max3421e. USB *Host* max3421e tersedia di pasar dalam bentuk modul yang kompatibel dengan *port* modul Arduino. Untuk dapat terkoneksi dengan mikrokontroller ESP32, diperlukan *routing* yang dirangkai di *prototype board* seperti terlihat pada Gambar 1 berikut:

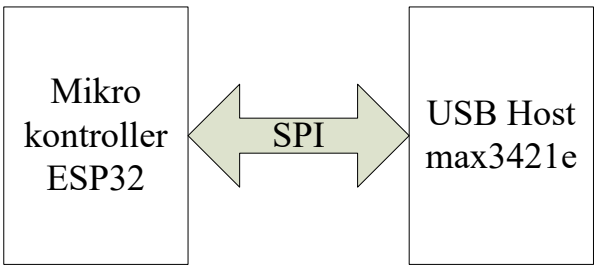


Gambar 1. *Routing* Mikrokontroller ESP32 dan USB Host max3421e

Setelah terkoneksi, dilakukan kegiatan pengecekan koneksi menggunakan Avometer. Kemudian dilanjutkan dengan pembuatan *software* yang akan ditanam di mikrokontroller ESP32. Pengembangan *software* ini menggunakan fasilitas yang sudah disediakan oleh pembuat mikrokontroller ESP32 yaitu ESP-IDF (IoT Development Framework.

2.1. Blok Diagram Sistem

Sistem yang akan dirancang terdiri atas mikrokontroller ESP32 dan juga USB *Host* max3421e digambarkan dalam blok diagram seperti pada Gambar 2 berikut:



Gambar 2. Blok diagram sistem

Mikrokontroller ESP32 dihubungkan dengan USB *Host* max3421e menggunakan SPI bus. Konfigurasi koneksi PIN yang dilakukan ditunjukkan dalam Tabel 1 berikut:

Tabel 1.Tabel Konfigurasi koneksi PIN

SPI Data	ESP32 Pin No	Max3421e Pin No
MOSI	23	4
MISO	19	1
SCK	18	3
CS	5	2

Komunikasi SPI yang digunakan pada penelitian ini menggunakan 4 jalur, yaitu MOSI, MISO, SCK dan CS. Mode SPI yang digunakan adalah mode 0 atau full duplex, yaitu baik *master* dan *slave* bisa mengirim data pada saat yang sama melalui jalur MOSI dan MISO. Selama komunikasi ini, data ditransmisikan secara bersamaan (bergeser secara serial ke bus MOSI / SDO) dan diterima (data di bus (MISO / SDI) disampel atau dibaca) [9].

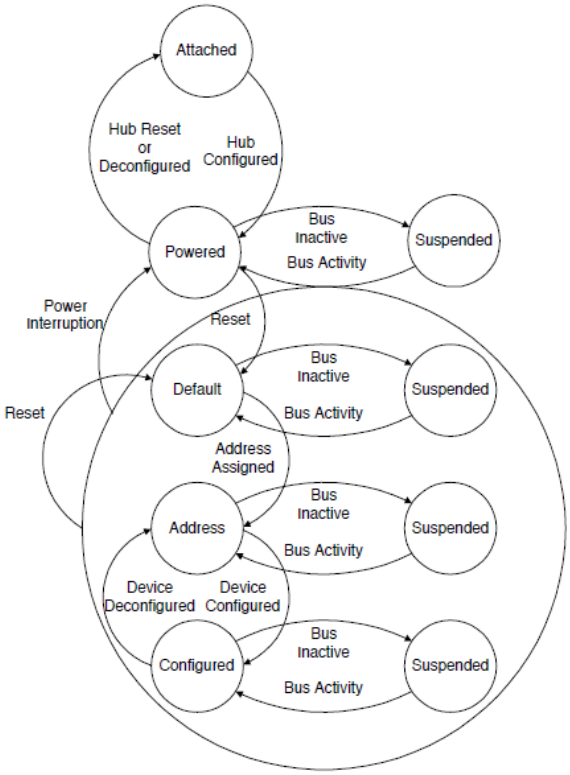
Untuk komunikasi SPI sendiri membutuhkan paling tidak tiga jalur, yakni MOSI, MISO, dan SCK. Melalui komunikasi ini data dapat saling dikirimkan baik antar mikrokontroler, maupun antara mikrokontroler dengan peripheral lainnya yang mendukung komunikasi dengan SPI [2].

2.2. Pengembangan *Software*

Pengembangan *software* memanfaatkan fasilitas ESP-IDF. Pengembangan ini dilakukan dengan cara menganalisis fitur SPI dari mikrokontroller ESP32 dan protokol USB. Data dikirimkan dari peripheral ke USB Host untuk dianalisis dan diketahui *unique address* nya. Dengan mengetahui *unique address* yang terdapat pada

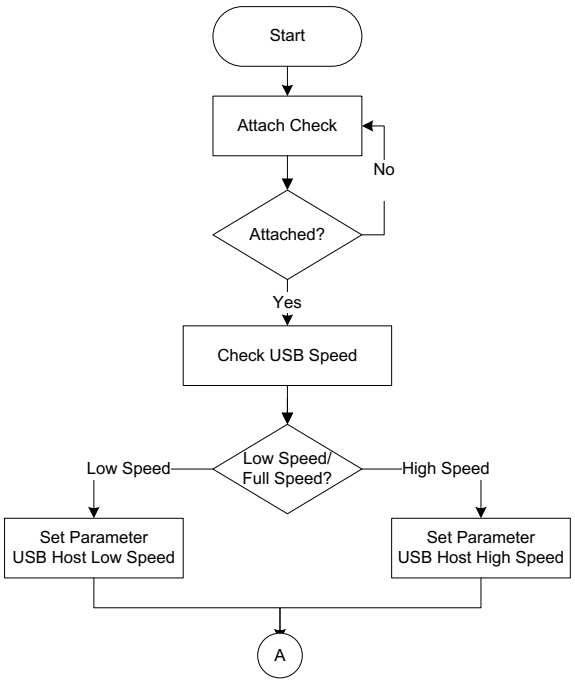
peripheral, komunikasi data antara *host* bisa dilakukan. Untuk memastikan bahwa pembacaan *device descriptor* dari *peripheral* adalah benar, perlu ditambahkan command di dalam *software*, sehingga pengecekan data tersebut dapat dilakukan di PC melalui kabel USB.

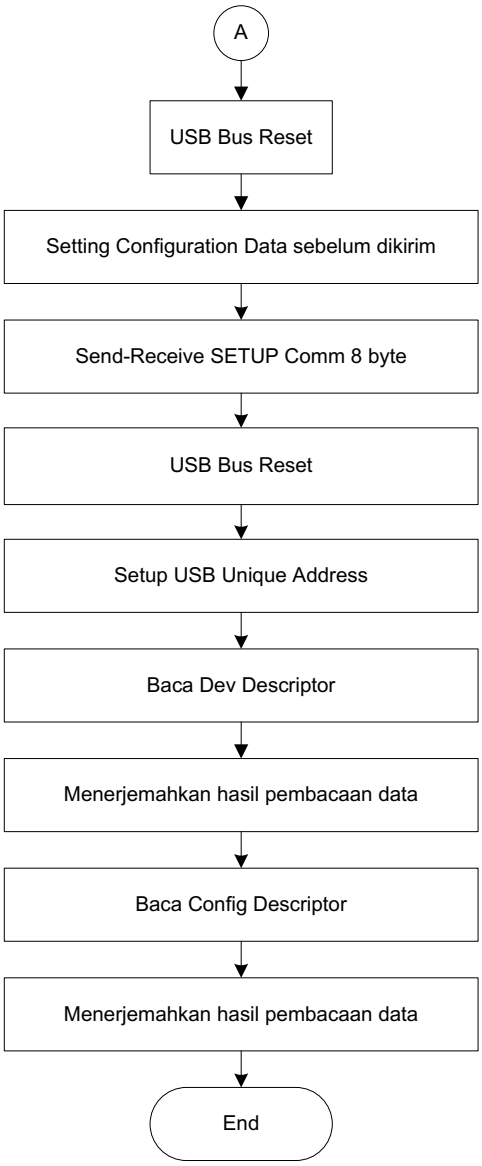
Pengembangan *software* dilakukan berdasarkan *Device State Diagram* dari USB Device Framework [10] seperti yang ditampilkan dalam Gambar 3 berikut:



Gambar 3. Blok diagram sistem

Adapun diagram alir dari *software* yang dikembangkan dari *Device State Diagram* diatas ditunjukkan dalam Gambar 4 berikut:



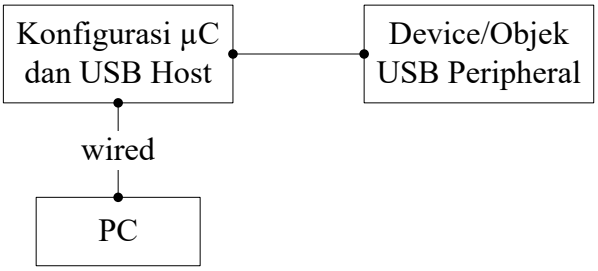


Gambar 4. Diagram alir *software*

Saat perangkat USB dipasang atau dilepas dari USB, host menggunakan proses yang disebut bus enumerasi untuk mengidentifikasi dan mengelola perubahan status perangkat yang diperlukan. Saat perangkat USB terhubung ke perangkat, langkah-langkah yang digambarkan pada Gambar 4 di atas dilakukan secara berurutan. Saat perangkat USB dilepas, hub kembali mengirimkan pemberitahuan ke *host*. Melepaskan perangkat berarti menonaktifkan *port* yang telah terpasang. Setelah menerima pemberitahuan pelepasan, *host* akan memperbarui informasi topologi lokalnya [10].

2.3. Tahapan Pengujian

Pengujian yang dilakukan pada penelitian ini adalah proses komunikasi menggunakan peralatan elektronik USB dengan kategori Low-Speed, Full-Speed dan High-Speed yaitu printer Epson L210 yang tidak dilengkapi dengan fasilitas WiFi, *flashdisk* SanDisk, *mouse* bluetooth Logitech dan juga External Hard Disk Drive (HDD). Peralatan yang dikembangkan akan mencoba melakukan proses *handshaking* dengan semua peralatan elektronik tersebut sebagai objek USB *peripheral*. Adapun konfigurasi proses pengujian yang dilakukan ditampilkan pada Gambar 5 berikut:



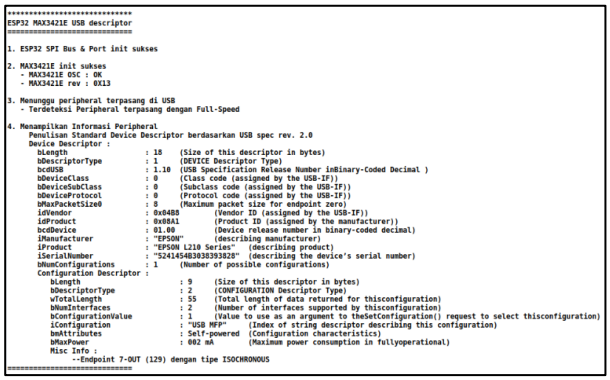
Gambar 5. Konfigurasi proses pengujian

3. Hasil dan Pembahasan

Pengujian terhadap sistem yang sedang dikembangkan dilakukan dengan menggunakan sistem operasi Linux untuk setiap *device* atau objek USB peripheral.

3.1. Pengujian dengan Printer Epson L210

Pengujian dilakukan dengan cara menghubungkan printer Epson L210 ke *port* USB Host max3421e dan PC ke mikrokontroller ESP32. Kemudian diambil data *device descriptor* yang dihasilkan dari koneksi tersebut. Adapun data *device descriptor* yang dihasilkan, ditampilkan pada Gambar 6 berikut:



Gambar 6. Hasil *device descriptor* dari printer Epson L210

Dalam gambar tersebut, ditunjukkan iManufacturer = EPSON dan iProduct = EPSON L210 Series. Hal ini menunjukkan bahwa *device descriptor* yang dihasilkan dari *software* yang dikembangkan dapat berjalan dengan benar. Sebagai pembanding, *device* printer Epson langsung dihubungkan ke PC dan dilihat *device descriptor* dari PC. Hasil yang didapatkan ditampilkan dalam Gambar 7 berikut:

```
Bus 001 Device 008: ID 04b8:08a1 Seiko Epson Corp.
Device Descriptor:
  bLength                18
  bDescriptorType         1
  bcdUSB               1.10
  bDeviceClass            0 (Defined at Interface level)
  bDeviceSubClass         0
  bDeviceProtocol         0
  bMaxPacketSize0         8
  idVendor           0x04b8 Seiko Epson Corp.
  idProduct          0x08a1
  bcdDevice            1.00
  iManufacturer           1 EPSON
  iProduct                2 EPSON L210 Series
  iSerial                 3 5241454B3038393828
  bNumConfigurations      1
  Configuration Descriptor:
    bLength                 9
    bDescriptorType         2
    wTotalLength           55
    bNumInterfaces          2
    bConfigurationValue     1
    iConfiguration          4 USB MFP
    bmAttributes         0xc0
      Self Powered
    MaxPower                2mA
```

Gambar 7. Hasil *device descriptor* dari PC

Dari Gambar 7 di atas, dapat dilihat bahwa perangkat USB yang sedang terhubung di PC adalah printer EPSON L210 Series, ditunjukkan dengan iManufacturer = EPSON dan iProduct = EPSON L210 Series.

3.2. Pengujian dengan *Flashdisk* SanDisk

Dengan cara yang sama dengan pengujian menggunakan printer Epson di atas, didapatkan hasil pengujian menggunakan *flashdisk* SanDisk seperti yang ditunjukkan dalam Gambar 8 berikut:

```
**************************
ESP32 MAX3421E USB descriptor
==========================

1. ESP32 SPI Bus & Port init sukses

2. MAX3421E init sukses
   - MAX3421E OSC : OK
   - MAX3421E rev : 0X13

3. Menunggu peripheral terpasang di USB
   - Terdeteksi Peripheral terpasang dengan Full-Speed

4. Menampilkan Informasi Peripheral
     Penulisan Standard Device Descriptor berdasarkan USB spec rev. 2.0
     Device Descriptor :
       bLength             : 18    (Size of this descriptor in bytes)
       bDescriptorType     : 1     (DEVICE Descriptor Type)
       bcdUSB              : 2.00  (USB Specification Release Number inBinary-Coded Decimal )
       bDeviceClass        : 0     (Class code (assigned by the USB-IF))
       bDeviceSubClass     : 0     (Subclass code (assigned by the USB-IF))
       bDeviceProtocol     : 0     (Protocol code (assigned by the USB-IF))
       bMaxPacketSize0     : 64    (Maximum packet size for endpoint zero)
       idVendor            : 0x0781        (Vendor ID (assigned by the USB-IF))
       idProduct           : 0x5567        (Product ID (assigned by the manufacturer))
       bcdDevice           : 01.00         (Device release number in binary-coded decimal)
       iManufacturer       : "SanDisk"     (describing manufacturer)
       iProduct            : "Cruzer Blade"        (describing product)
       iSerialNumber       : "4C530000081026114422"        (describing the device's serial number)
       bNumConfigurations  : 1     (Number of possible configurations)
       Configuration Descriptor :
          bLength                 : 9     (Size of this descriptor in bytes)
          bDescriptorType         : 2     (CONFIGURATION Descriptor Type)
          wTotalLength            : 32    (Total length of data returned for thisconfiguration)
          bNumInterfaces          : 1     (Number of interfaces supported by thisconfiguration)
          bConfigurationValue     : 1     (Value to use as an argument to theSetConfiguration() request to select thisconfiguration)
          iConfiguration          : "No Config string"    (Index of string descriptor describing this configuration)
          bmAttributes            : Remote Wakeup/Bus-Powered     (Configuration characteristics)
          bMaxPower               : 200 mA        (Maximum power consumption in fullyoperational)
          Misc Info :
               Interface 0
               Alternate Setting 0
               --Endpoint 1-IN  (64) dengan tipe BULK
               --Endpoint 2-OUT (64) dengan tipe BULK
==========================
```

Gambar 8. Hasil *device descriptor* dari flashdisk SanDisk

Dalam gambar tersebut, ditunjukkan iManufacturer = SanDisk dan iProduct = Cruzer Blade. Hal ini menunjukkan bahwa *device descriptor* yang dihasilkan dari *software* yang dikembangkan dapat berjalan dengan benar. Dalam Gambar 9 berikut ditunjukkan *device descriptor* berupa idVendor = SanDisk Corp. dan idProduct = Cruzer Blade yang diambil dari PC ketika *device flashdisk* SanDisk langsung dihubungkan ke PC.

```
Bus 001 Device 007: ID 0781:5567 SanDisk Corp. Cruzer Blade
Couldn't open device, some information will be missing
Device Descriptor:
  bLength                18
  bDescriptorType         1
  bcdUSB               2.00
  bDeviceClass            0 (Defined at Interface level)
  bDeviceSubClass         0
  bDeviceProtocol         0
  bMaxPacketSize0        64
  idVendor           0x0781 SanDisk Corp.
  idProduct          0x5567 Cruzer Blade
  bcdDevice            1.00
  iManufacturer           1
  iProduct                2
  iSerial                 3
  bNumConfigurations      1
  Configuration Descriptor:
    bLength                 9
    bDescriptorType         2
    wTotalLength           32
    bNumInterfaces          1
    bConfigurationValue     1
    iConfiguration          0
    bmAttributes         0x80
      (Bus Powered)
    MaxPower              200mA
```

Gambar 9. Hasil *device descriptor* dari PC

3.3. Pengujian dengan *Mouse* Bluetooth Logitech

Dengan cara yang sama dengan pengujian menggunakan printer Epson dan *flashdisk* di atas, didapatkan hasil pengujian menggunakan *mouse* Bluetooth Logitech seperti yang ditunjukkan dalam Gambar 10 berikut:

```
**************************
ESP32 MAX3421E USB descriptor
==========================

1. ESP32 SPI Bus & Port init sukses

2. MAX3421E init sukses
   - MAX3421E OSC : OK
   - MAX3421E rev : 0X13

3. Menunggu peripheral terpasang di USB
   - Terdeteksi Peripheral terpasang dengan Full-Speed

4. Menampilkan Informasi Peripheral
     Penulisan Standard Device Descriptor berdasarkan USB spec rev. 2.0
     Device Descriptor :
       bLength             : 18    (Size of this descriptor in bytes)
       bDescriptorType     : 1     (DEVICE Descriptor Type)
       bcdUSB              : 2.00  (USB Specification Release Number inBinary-Coded Decimal )
       bDeviceClass        : 0     (Class code (assigned by the USB-IF))
       bDeviceSubClass     : 0     (Subclass code (assigned by the USB-IF))
       bDeviceProtocol     : 0     (Protocol code (assigned by the USB-IF))
       bMaxPacketSize0     : 8     (Maximum packet size for endpoint zero)
       idVendor            : 0x046D        (Vendor ID (assigned by the USB-IF))
       idProduct           : 0xC534        (Product ID (assigned by the manufacturer))
       bcdDevice           : 29.01         (Device release number in binary-coded decimal)
       iManufacturer       : "Logitech"    (describing manufacturer)
       iProduct            : "USB Receiver"        (describing product)
       Tidak memiliki Serial Number
       bNumConfigurations      : 1         (Number of possible configurations)
       Configuration Descriptor :
          bLength                 : 9     (Size of this descriptor in bytes)
          bDescriptorType         : 2     (CONFIGURATION Descriptor Type)
          wTotalLength            : 59    (Total length of data returned for thisconfiguration)
          bNumInterfaces          : 2     (Number of interfaces supported by thisconfiguration)
          bConfigurationValue     : 1     (Value to use as an argument to theSetConfiguration() request to select thisconfiguration)
          iConfiguration          : "RQR29.01_B0016"      (Index of string descriptor describing this configuration)
          bmAttributes            : Remote Wakeup/Bus-Powered     (Configuration characteristics)
          bMaxPower               : 098 mA        (Maximum power consumption in fullyoperational)
          Misc Info :
               --Endpoint 2-IN  (20) dengan tipe INTERRUPT dengan interval polling 2 msec.
==========================
```

Gambar 10. Hasil *device descriptor* dari *mouse* bluetooth Logitech

Dalam gambar tersebut, ditunjukkan iManufacturer = Logitech dan iProduct = USB Receiver. Hal ini menunjukkan bahwa *device descriptor* yang dihasilkan dari *software* yang dikembangkan dapat berjalan dengan benar. Sebagai pembanding, *device mouse* Bluetooth Logitech langsung dihubungkan ke PC dan dilihat *device descriptor* dari PC. Hasil yang didapatkan ditampilkan dalam Gambar 11 berikut:

```
Bus 001 Device 002: ID 046d:c534 Logitech, Inc. Unifying Receiver
Couldn't open device, some information will be missing
Device Descriptor:
  bLength                18
  bDescriptorType         1
  bcdUSB               2.00
  bDeviceClass            0 (Defined at Interface level)
  bDeviceSubClass         0
  bDeviceProtocol         0
  bMaxPacketSize0         8
  idVendor           0x046d Logitech, Inc.
  idProduct          0xc534 Unifying Receiver
  bcdDevice           29.01
  iManufacturer           1
  iProduct                2
  iSerial                 0
  bNumConfigurations      1
  Configuration Descriptor:
    bLength                 9
    bDescriptorType         2
    wTotalLength           59
    bNumInterfaces          2
    bConfigurationValue     1
    iConfiguration          4
    bmAttributes         0xa0
      (Bus Powered)
      Remote Wakeup
    MaxPower                98mA
```

Gambar 11. Hasil *device descriptor* dari PC

Dari Gambar 11 di atas, ditunjukkan *device descriptor* berupa idVendor = Logitech, Inc dan idProduct = Unifying Receiver yang diambil dari PC ketika *device mouse* Bluetooth Logitech langsung dihubungkan ke PC.

3.4. Pengujian dengan External Hard Disk Drive (HDD) Logitec

Dengan cara pengujian yang sama dengan ketiga device di atas, didapatkan hasil pengujian *device descriptor* menggunakan *external* HDD Logitec berkapasitas 150 GB seperti yang ditunjukkan dalam Gambar 12 berikut ini:

```
****************************
ESP32 MAX3421E USB descriptor
****************************

1. ESP32 SPI Bus & Port init sukses

2. MAX3421E init sukses
   - MAX3421E OSC : OK
   - MAX3421E rev : 0X13

3. Menunggu peripheral terpasang di USB
   - Terdeteksi Peripheral terpasang dengan Full-Speed

4. Menampilkan Informasi Peripheral
     Penulisan Standard Device Descriptor berdasarkan USB spec rev. 2.0
     Device Descriptor :
       bLength             : 18    (Size of this descriptor in bytes)
       bDescriptorType     : 1     (DEVICE Descriptor Type)
       bcdUSB              : 2.00  (USB Specification Release Number inBinary-Coded Decimal )
       bDeviceClass        : 0     (Class code (assigned by the USB-IF))
       bDeviceSubClass     : 0     (Subclass code (assigned by the USB-IF))
       bDeviceProtocol     : 0     (Protocol code (assigned by the USB-IF))
       bMaxPacketSize0     : 64    (Maximum packet size for endpoint zero)
       idVendor            : 0x0789        (Vendor ID (assigned by the USB-IF))
       idProduct           : 0x00CC        (Product ID (assigned by the manufacturer))
       bcdDevice           : 01.17         (Device release number in binary-coded decimal)
       iManufacturer       : "Logitec Corp.   "    (describing manufacturer)
       iProduct            : "LHD USB Device  "    (describing product)
       iSerialNumber       : "0170109000000F71"    (describing the device's serial number)
       bNumConfigurations  : 1     (Number of possible configurations)
       Configuration Descriptor :
          bLength                 : 9     (Size of this descriptor in bytes)
          bDescriptorType         : 2     (CONFIGURATION Descriptor Type)
          wTotalLength            : 32    (Total length of data returned for thisconfiguration)
          bNumInterfaces          : 1     (Number of interfaces supported by thisconfiguration)
          bConfigurationValue     : 1     (Value to use as an argument to theSetConfiguration() request to select thisconfiguration)
          iConfiguration          : "No Config string"    (Index of string descriptor describing this configuration)
          bmAttributes            : Self-powered  (Configuration characteristics)
          bMaxPower               : 002 mA        (Maximum power consumption in fullyoperational)
          Misc Info :
               Interface 0
               Alternate Setting 0
               --Endpoint 1-IN  (64) dengan tipe BULK
               --Endpoint 2-OUT (64) dengan tipe BULK
************************
```

Gambar 12. Hasil *device descriptor* dari external HDD Logitec

4. Kesimpulan

Dari penelitian yang telah dilakukan, didapatkan hasil bahwa antarmuka antara mikrokontroller ESP32 dengan USB Host max3421e dapat dilakukan dengan baik. Proses *handshake* yang terjadi antara sistem dengan USB device berkategori Low-Speed dan Full-Speed seperti printer, flashdisk, Bluetooth mouse dan external harddisk dapat dilakukan dengan baik, serta data *device descriptor* dapat dibaca dengan benar. Pengembangan lebih lanjut dapat dilakukan agar sistem mampu melakukan proses *handshake* dengan USB *device* berkategori High-Speed. Selain itu, dengan memanfaatkan fasilitas IoT di mikrokontroller, diharapkan sistem dapat mengakses peralatan elektronik dengan teknologi lama dan *entry level* secara *wireless* melalui WiFi.

Daftar Rujukan

[1] T. Darmanto and H. Krisma, “Implementasi Teknologi IOT Untuk Pengontrolan Peralatan Elektronik Rumah Tangga Berbasis Android,” *J. Tek. Inform. Unika St. Thomas*, vol. 04, pp. 1–12, 2019.

[2] R. Susana, M. Ichwan, and S. A. L. Phard, “Penerapan Metoda Serial Peripheral Interface (SPI) pada Rancang Bangun Data Logger berbasis SD card,” *J. Elkomika*, vol. 4, no. 2, pp. 208–227, 2016.

[3] A. Astudillo, P. Munoz, F. Alvarez, and E. Rosero, “Altitude and Attitude Cascade Controller for a Smartphone-based Quadcopter,” in *2017 International Conference on Unmanned Aircraft Systems, ICUAS 2017*, 2017, pp. 1447–1454, doi: 10.1109/ICUAS.2017.7991400.

[4] X. Li, H. Yu, Y. Yew, H. Liu, and M. Hong, “A Portable Optical Sensing System for Rapid Detection of Fluorescence Spectra,” *Guangdian Gongcheng/Opto-Electronic Eng.*, vol. 44, no. 5, pp. 483–487, 2017, doi: 10.3969/j.issn.1003-501X.2017.05.002.

[5] M. Iqbal, T. W. Widodo, and B. A. A. Sumbodo, “Sistem Pengendali Pengambilan Gambar Pada Kamera DSLR Melalui Protokol PTP 1,” *IJEIS*, vol. 6, no. 2, pp. 117–128, 2016.

[6] A. Novriadi, “Perancangan Pengontrolan Overhead Crane Menggunakan Kabel dan Nirkabel Berbasis Arduino sangat banyak ditemukan di berbagai dunia bekerja mengangkut bahan material Agar terhindarnya dari kecelakaan penggunaan yang digunakan adalah Joystick PlayStation3 Ar,” *J. Teknol. Terpadu*, vol. 7, no. 2, pp. 76–84, 2019.

[7] E. Systems, “ESP32 Series Datasheet,” 2021.

[8] I. Circuits, “Data sheet,” no. September 1993, 1998.

[9] P. Dhaker, “Introduction to SPI Interface,” *Analog Dialogue*, no. September. pp. 1–5, 2018, [Online]. Available: https://www.analog.com/media/en/analog-dialogue/volume-52/number-3/introduction-to-spi-interface.pdf.

[10] U. S. B. D. States and V. D. States, “Chapter 9 USB Device Framework,” in *Universal Serial Bus Specification Revision 2.0*, pp. 239–274.